```
RRRRRRRRRRRR    MMM         MMM    SSSSSSSSSSSS
RRRRRRRRRRRR    MMM         MMM    SSSSSSSSSSSS
RRRRRRRRRRRR    MMM         MMM    SSSSSSSSSSSS
RRR        RRR  MMMMMM    MMMMM  SSS
RRR        RRR  MMMMMM    MMMMM  SSS
RRR        RRR  MMMMMM    MMMMM  SSS
RRR        RRR  MMM  MMM   MMM   SSS
RRR        RRR  MMM   MMM  MMM   SSS
RRRRRRRRRRRR    MMM         MMM    SSSSSSSSS
RRRRRRRRRRRR    MMM         MMM    SSSSSSSSS
RRRRRRRRRRRR    MMM         MMM    SSSSSSSSS
RRR   RRR       MMM         MMM            SSS
RRR   RRR       MMM         MMM            SSS
RRR   RRR       MMM         MMM            SSS
RRR     RRR     MMM         MMM            SSS
RRR     RRR     MMM         MMM            SSS
RRR     RRR     MMM         MMM            SSS
RRR       RRR   MMM         MMM    SSSSSSSSSSSS
RRR       RRR   MMM         MMM    SSSSSSSSSSSS
RRR       RRR   MMM         MMM    SSSSSSSSSSSS
```

```
RRRRRRR    MM      MM        11       UU        UU  PPPPPPPP  DDDDDDD      AAAAAA    TTTTTTTTTT  EEEEEEEEEE
RRRRRRR    MM      MM        11       UU        UU  PPPPPPPP  DDDDDDD      AAAAAA    TTTTTTTTTT  EEEEEEEEEE
RR    RR   MMMM  MMMM      1111       UU        UU  PP    PP  DD    DD   AA    AA        TT      EE
RR    RR   MMMM  MMMM      1111       UU        UU  PP    PP  DD    DD   AA    AA        TT      EE
RR    RR   MM MM  MM         11       UU        UU  PP    PP  DD    DD   AA    AA        TT      EE
RR    RR   MM MM  MM         11       UU        UU  PP    PP  DD    DD   AA    AA        TT      EE
RRRRRRR    MM      MM        11       UU        UU  PPPPPPPP  DD    DD   AA    AA        TT      EEEEEEEE
RRRRRRR    MM      MM        11       UU        UU  PPPPPPPP  DD    DD   AA    AA        TT      EEEEEEEE
RR  RR     MM      MM        11       UU        UU  PP        DD    DD   AAAAAAAAAA      TT      EE
RR  RR     MM      MM        11       UU        UU  PP        DD    DD   AAAAAAAAAA      TT      EE
RR    RR   MM      MM        11       UU        UU  PP        DD    DD   AA    AA        TT      EE       ....
RR    RR   MM      MM        11       UU        UU  PP        DD    DD   AA    AA        TT      EE       ....
RR    RR   MM      MM    111111   UUUUUUUUUU    PP        DDDDDDD    AA    AA        TT      EEEEEEEEEE   ....
RR    RR   MM      MM    111111   UUUUUUUUUU    PP        DDDDDDD    AA    AA        TT      EEEEEEEEEE   ....
```

```
LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL    IIIIII      SSSSSSS
LLLLLLLLLL    IIIIII      SSSSSSS
```

RM1UPDATE
V04-000

SEQUENTIAL SPECIFIC UPDATE

G 14

16-SEP-1984 00:58:37   VAX/VMS Macro V04-00
5-SEP-1984 16:23:54   [RMS.SRC]RM1UPDATE.MAR;1

Page   1
       (1)

```
0000      1              $BEGIN  RM1UPDATE,000,RM$RMS1,<SEQUENTIAL SPECIFIC UPDATE>
0000      2
0000      3      ;
0000      4      ;*******************************************************************************
0000      5      ;*                                                                             *
0000      6      ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                    *
0000      7      ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                     *
0000      8      ;*  ALL RIGHTS RESERVED.                                                       *
0000      9      ;*                                                                             *
0000     10      ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED      *
0000     11      ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE      *
0000     12      ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER      *
0000     13      ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY      *
0000     14      ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY      *
0000     15      ;*  TRANSFERRED.                                                               *
0000     16      ;*                                                                             *
0000     17      ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE      *
0000     18      ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT      *
0000     19      ;*  CORPORATION.                                                               *
0000     20      ;*                                                                             *
0000     21      ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS      *
0000     22      ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
0000     23      ;*                                                                             *
0000     24      ;*                                                                             *
0000     25      ;*******************************************************************************
0000     26      ;
```

RM1UPDATE
V04-000

SEQUENTIAL SPECIFIC UPDATE

H 14

16-SEP-1984 00:58:37   VAX/VMS Macro V04-00   Page 2
5-SEP-1984 16:23:54   [RMS.SRC]RM1UPDATE.MAR;1   (2)

```
0000   28  ;++
0000   29  ; Facility: rms32
0000   30  ;
0000   31  ; Abstract:
0000   32  ;           this module provides sequential file organization
0000   33  ;           specific processing for the $update function.
0000   34  ;
0000   35  ;
0000   36  ; Environment:
0000   37  ;           star processor running starlet exec.
0000   38  ;
0000   39  ; Author:    l f laverdure,   creation date: 14-JUL-1977
0000   40  ;
0000   41  ; Modified By:
0000   42  ;
0000   43  ;     V03-010 JEJ0051         J E Johnson            07-Aug-1984
0000   44  ;             Back out JEJ0049 due to some unexplained side effects.
0000   45  ;
0000   46  ;     V03-009 JEJ0049         J E Johnson            23-Jul-1984
0000   47  ;             Alter the logic in BLDREC to force a flush of the current
0000   48  ;             buffer if it is exactly filled by the record instead of
0000   49  ;             waiting for the next operation to force it out.
0000   50  ;
0000   51  ;     V03-008 TSK0001         Tamar Krichevsky        9-Dec-1983
0000   52  ;             Add support for BI journaling and recovery. First, make sure
0000   53  ;             the buffer is always filled. That is, no optimizations are
0000   54  ;             done (such as skipping reads or doing short reads) when the
0000   55  ;             file is being BI journaled. We need the whole record in the
0000   56  ;             buffer, so that it can be copied to the journal entry.
0000   57  ;             Second, if BI recovery is occuring, do not append any missing
0000   58  ;             STREAM terminators to the end of a record. The data put back
0000   59  ;             in the file must be exactly the same as that which was taken
0000   60  ;             out. Appending missing terminators to the end record may
0000   61  ;             overwrite data.
0000   62  ;
0000   63  ;     V03-007 TSK0002         Tamar Krichevsky       22-Jun-1983
0000   64  ;             Fix broken branch to RM$SEQJNL.
0000   65  ;
0000   66  ;     V03-006 TSK0001         Tamar Krichevsky       21-Jun-1983
0000   67  ;             Add support for journaling $UPDATE operations.
0000   68  ;
0000   69  ;     V03-005 TMK0001         Todd M. Katz           27-Dec-1982
0000   70  ;             Clear the bit IRB$V_FIND_LAST as soon as RM$UPDATE1 is entered.
0000   71  ;
0000   72  ;     V03-004 KPL0001         Peter Lieberwirth      20-Dec-1982
0000   73  ;             Fix a bug introduced some time during V3.0 development that
0000   74  ;             broke updating when the multi-block count is 1 and the record
0000   75  ;             happened to be just the right length (like 256 for example).
0000   76  ;             Improve the commentary where the magic is.
0000   77  ;
0000   78  ;     V03-003 KBT0419         Keith B. Thompson      30-Nov-1982
0000   79  ;             Change ifb$w_devbufsiz to ifb$l_devbufsiz
0000   80  ;
0000   81  ;     V03-002 KBT0150         Keith B. Thompson      20-Aug-1982
0000   82  ;             Reorganize psects
0000   83  ;
0000   84  ;     V03-001 KBT0090         Keith B. Thompson      13-Jul-1982
```

```
0000      85 ;              Clean up psects
0000      86 ;
0000      87 ;      V02-015 TMK0034         Todd M. Katz    22-Dec-1981
0000      88 ;              Fix a broken branch by change a BRW RM$PUT_UNIT_REC
0000      89 ;              to a JMP.
0000      90 ;
0000      91 ;      V02-014 RAS0030         Ron Schaefer    25-Aug-1981
0000      92 ;              Fix broken branch caused by _device parsing.
0000      93 ;
0000      94 ;      V02-013 RAS0028         Ron Schaefer    20-Aug-1981
0000      95 ;              Change FAB$C_STM11 to FAB$C_STM.
0000      96 ;
0000      97 ;      V02-012 RAS0016         Ron Schaefer    6-Aug-1981
0000      98 ;              Add stream file support.
0000      99 ;
0000     100 ;      V02-011 REFORMAT        R A Schaefer    25-Jul-1980
0000     101 ;              Reformat the source
0000     102 ;
0000     103 ;      V010    JAK0040         J A Krycka      15-FEB-1980
0000     104 ;              fix bug in network $update.
0000     105 ;
0000     106 ;--
0000     107 ;
0000     108
```

J 14

RM1UPDATE   SEQUENTIAL SPECIFIC UPDATE   16-SEP-1984 00:58:37 VAX/VMS Macro V04-00   Page   4
V04-000   DECLARATIONS   5-SEP-1984 16:23:54 [RMS.SRC]RM1UPDATE.MAR;1   (3)

```
                      0000   110             .SBTTL  DECLARATIONS
                      0000   111
                      0000   112  ;
                      0000   113  ; Include Files:
                      0000   114  ;
                      0000   115
                      0000   116  ; Macros:
                      0000   117  ;
                      0000   118
                      0000   119
                      0000   120             $IFBDEF
                      0000   121             $DEVDEF
                      0000   122             $FABDEF
                      0000   123             $RABDEF
                      0000   124             $IRBDEF
                      0000   125             $BDBDEF
                      0000   126             $RMSDEF
                      0000   127             $RJRDEF
                      0000   128
                      0000   129  ;
                      0000   130  ; Equated Symbols:
                      0000   131  ;
                      0000   132
              0000000A 0000  133             LF=10
              0000000D 0000  134             CR=13
                      0000   135
                      0000   136  ;
                      0000   137  ; Own Storage:
                      0000   138  ;
                      0000   139
                      0000   140  ;
                      0000   141  ; Stream format default terminators (DFT)
                      0000   142  ;
                      0000   143  STM_FMT_DFT:
        00 0A 0D 02  0000   144             .BYTE   2, CR, LF, 0                 ; FAB$C_STM
        00 00 0A 01  0004   145             .BYTE   1, LF, 0, 0                  ; FAB$C_STMLF
        00 00 0D 01  0008   146             .BYTE   1, CR, 0, 0                  ; FAB$C_STMCR
                      000C   147
```

RM1UPDATE   SEQUENTIAL SPECIFIC UPDATE   16-SEP-1984 00:58:37   VAX/VMS Macro V04-00   Page 5
V04-000   RMSUPDATE1 - HIGH LEVEL SEQUENTIAL $UPDA 5-SEP-1984 16:23:54   [RMS.SRC]RM1UPDATE.MAR;1   (4)

K 14

```
000C     149              .SBTTL   RMSUPDATE1 - HIGH LEVEL SEQUENTIAL $UPDATE
000C     150
000C     151  ;++
000C     152  ; RMSUPDATE1:                   high level sequential $update
000C     153  ; RMSUPDATE_ALT:                alternate entry point for put random
000C     154  ;
000C     155  ; this module performs the following functions:
000C     156  ;
000C     157  ;        1.  calls rm$putsetup1 to perform various setups
000C     158  ;        2.  verifies that the file is on disk, that we have a
000C     159  ;            current record, and that the size is not changing
000C     160  ;        3.  reads the block containing the record if necessary
000C     161  ;        4.  moves the updated record to the block buffer setting
000C     162  ;            the buffer dirty flag
000C     163  ;
000C     164  ; Calling sequence:
000C     165  ;
000C     166  ;        entered via case branch from rm$update at rm$update1.
000C     167  ;
000C     168  ; Input Parameters:
000C     169  ;
000C     170  ;        r11     impure area address
000C     171  ;        r10     ifab addr
000C     172  ;        r9      irab addr
000C     173  ;        r8      rab addr
000C     174  ;
000C     175  ; Implicit Inputs:
000C     176  ;
000C     177  ;        the contents of the rab and related irab and ifab.
000C     178  ;
000C     179  ; Output Parameters:
000C     180  ;
000C     181  ;        r7 thru r1          destroyed
000C     182  ;        r0                  status
000C     183  ;
000C     184  ; Implicit Outputs:
000C     185  ;
000C     186  ;        various fields of the rab are filled in to reflect
000C     187  ;        the status of the operation (see functional spec
000C     188  ;        for details).
000C     189  ;
000C     190  ;        the irab is similarly updated.
000C     191  ;
000C     192  ; Completion Codes:
000C     193  ;
000C     194  ;        standard rms (see functional spec).
000C     195  ;
000C     196  ; Side Effects:
000C     197  ;
000C     198  ;        none
000C     199  ;--
000C     200
000C     201
000C     202  RMSUPDATE1::
000C     203              $TSTPT   UPDATE1
0012     204              CSB      #IRB$V_FIND_LAST,(R9)   ; last operation is no longer $FIND
FFE7'  30   0016     205              BSBW     RM$PUTSETUPT            ; perform various update setups
```

```
          32 50   E9   0019   206              BLBC      R0,UPDERR
       3A 6A 3E   E0   001C   207              BBS       #IFB$V_DAP,(R10),NTUPD   ; branch if network file access
                        0020   208
                        0020   209  ;
                        0020   210  ; make various legal operation checks
                        0020   211  ;
                        0020   212
          62 A9   B5   0020   213              TSTW      IRB$W_CSIZ(R9)           ; was there a current rec
             1E   13   0023   214              BEQL      ERRCUR                   ; branch if no
          50 AA   91   0025   215              CMPB      IFB$B_RFMORG(R10),-      ; stream format?
             04        0028   216                        #FAB$C_STM
             06   1E   0029   217              BGEQU     10$                      ; no size check for stream
       62 A9 51   B1   002B   218              CMPW      R1,IRB$W_CSIZ(R9)        ; new size = current rec size?
             0B   12   002F   219              BNEQ      ERRRSZ                   ; branch if not
       52 6A 1C   E0   0031   220  10$:         BBS       #DEV$V_RND,IFB$L_PRIM_DEV(R10),UPDATE ; branch if disk
                        0035   221
                        0035   222  ;
                        0035   223  ;  handle errors
                        0035   224  ;
                        0035   225
                        0035   226              RMSERR    IOP                      ; device not disk
             12   11   003A   227              BRB       UPDERR
                        003C   228
                        003C   229  ERRRSZ:
                        003C   230              RMSERR    RSZ                      ; record size change attempted
             0B   11   0041   231              BRB       UPDERR
                        0043   232
                        0043   233  ERRCUR:
                        0043   234              RMSERR    CUR                      ; no current record
             04   11   0048   235              BRB       UPDERR
                        004A   236
                        004A   237  UPDERR_RSTNRP:
       40 A9 8E   7D   004A   238              MOVQ      (SP)+,IRB$L_NRP_VBN(R9)  ; restore nrp
                        004E   239                                                ; and fall thru to upderr1
                        004E   240  UPDERR:
          62 A9   B4   004E   241              CLRW      IRB$W_CSIZ(R9)           ; indicate no current record
          10 A8   D4   0051   242              CLRL      RAB$W_RFA(R8)            ; zero rfa
          14 A8   B4   0054   243              CLRW      RAB$W_RFA+4(R8)
          FFA6'  31   0057   244              BRW       RMS$EXRMS
                        005A   245
                        005A   246  ;
                        005A   247  ;  perform network update function
                        005A   248  ;
                        005A   249
                        005A   250  NTUPD:
                        005A   251              SSB       #IRB$V_UPDATE,(R9)       ; mark this as an update function
          FF9F'  30   005E   252              BSBW      RMS$PUT_UNIT_REC         ; join network $put code
                        0061   253
                        0061   254  ;
                        0061   255  ; set cache read flags approriately based upon the situation
                        0061   256  ;
                        0061   257  ; The idea here is that if we're writing an entire block (the record fills
                        0061   258  ; up an entire block or more) we don't have to read the block in, because
                        0061   259  ; it will be totally overwritten anyway.  However, if the record does not fill
                        0061   260  ; an entire block (if there are records before or after the record to update
                        0061   261  ; in the same block) then the block must be read in.
                        0061   262  ;
```

RM1UPDATE
V04-000

M 14
SEQUENTIAL SPECIFIC UPDATE   16-SEP-1984 00:58:37 VAX/VMS Macro V04-00   Page   7
RM$UPDATE1 - HIGH LEVEL SEQUENTIAL $UPDA 5-SEP-1984 16:23:54 [RMS.SRC]RM1UPDATE.MAR;1   (4)

```
                            0061   263 ; If     the record to be updated starts on a block boundary and either ends on
                            0061   264 ;        a block boundary or the end does not fit in the buffer
                            0061   265 ; Then
                            0061   266 ;          r3 is set to 2 to indicate a short read (r2 set to # of blocks to read)
                            0061   267 ; Else
                            0061   268 ;          r3 is cleared indicating a full buffer is to be read
                            0061   269
                            0061   270 NOFIT:                                      ; all blocks of record don't fit in buffer
            4C A9   B5      0061   271         TSTW    IRB$W_RP_OFF(R9)            ; is the start offset = 0?
               08   13      0064   272         BEQL    NOREAD                      ; branch if yes (no read required)
                            0066   273 READ_FIRST:
               52   B4      0066   274         CLRW    R2                          ; only the 1st blk needs to be read
               42   11      0068   275         BRB     GETBLK
                            006A   276
                            006A   277 ;
                            006A   278 ; check for ending offset = 0 and if so omit entire read
                            006A   279 ;
                            006A   280
               50   B5      006A   281 CHKEND: TSTW    R0                          ; is end offset 0?
               3E   12      006C   282         BNEQ    GETBLK                      ; branch if not (must read)
         03 50 AA   91      006E   283 NOREAD: CMPB    IFB$B_RFMORG(R10),#FAB$C_VFC ; rfm = vfc?
               38   13      0072   284         BEQL    GETBLK                      ; branch if yes - can't optimize
                            0074   285
                            0074   286 ;
                            0074   287 ; entry point for put past current eof block
                            0074   288 ;
                            0074   289
                            0074   290 NOREAD1:
         53    01   D0      0074   291         MOVL    #1,R3                       ; flag read not needed
               33   11      0077   292         BRB     GETBLK
                            0079   293
                            0079   294 ;
                            0079   295 ; entry point from put random
                            0079   296 ;
                            0079   297 ; check for past eof block implying no read required
                            0079   298 ;
                            0079   299
                            0079   300 RM$UPDATE_ALT::
      74 AA 48 A9   D1      0079   301         CMPL    IRB$L_RP_VBN(R9),IFB$L_EBK(R10) ; past eof block?
               07   1F      007E   302         BLSSU   UPDATE                      ; branch if not
               F2   1A      0080   303         BGTRU   NOREAD1                     ; branch if yes
            5C AA   B5      0082   304         TSTW    IFB$W_FFB(R10)              ; any data in block?
               ED   13      0085   305         BEQL    NOREAD1            .        ; branch if none
```

RM1UPDATE
V04-000

N 14

SEQUENTIAL SPECIFIC UPDATE   16-SEP-1984 00:58:37 VAX/VMS Macro V04-00   Page 8
RMSUPDATE1 - HIGH LEVEL SEQUENTIAL SUPDA 5-SEP-1984 16:23:54 [RMS.SRC]RM1UPDATE.MAR;1   (6)

```
                                 0087   307
                                 0087   308
                                 0087   309  ; current register contents:
                                 0087   310  ;
                                 0087   311  ;      r11-r8  normal rms
                                 0087   312  ;      r6      record data length in bytes
                                 0087   313  ;      r5      record data address
                                 0087   314  ;      r1      total record size including overhead bytes
                                 0087   315  ;
                                 0087   316
                                 0087   317  UPDATE:
                                 0087   318
                                 0087   319  ; compute # of blocks to be read in if necessary
                                 0087   320  ;
                                 0087   321  ;
                                 0087   322
                  53   02   D0   0087   323          MOVL    #2,R3                       ; set flag for read required,
                                 008A   324                                              ;   explicit # of blocks
      50   51   4C A9   A1       008A   325          ADDW3   IRB$W_RP_OFF(R9),R1,R0      ; get end offset
                                 008F   326  ; Note that:    R0      - offset from buffer start to end of record+1
                                 008F   327  ;              RP_OFF  - offset from buffer start to start of record
                                 008F   328  ;
                                 008F   329  ;
            52   50   01   A3    008F   330          SUBW3   #1,R0,R2                    ; get actual end
      52   52   07   09   EF     0093   331          EXTZV   #9,#7,R2,R2                 ; get number of blocks - 1 in R2
            50 FE00 8F   AA      0098   332          BICW2   #^XFE00,R0                  ; offset in last block
            55 A9   52   91      009D   333          CMPB    R2,IRB$B_MBC(R9)            ; all blocks fit in buffer?
                       BE   1A   00A1   334          BGTRU   NOFIT                       ; branch if not
                                 00A3   335
                                 00A3   336  ;
                                 00A3   337  ; all blocks containing the desired record fit in the buffer
                                 00A3   338  ; check to see if either starting or ending offset is zero allowing
                                 00A3   339  ; for a short or null read
                                 00A3   340  ;
                                 00A3   341
                  4C A9   B5     00A3   342          TSTW    IRB$W_RP_OFF(R9)            ; start offset = 0?
                       C2   13   00A6   343          BEQL    CHKEND                      ; branch if yes
                       50   B5   00A8   344          TSTW    R0                          ; end offset = 0?
                       BA   13   00AA   345          BEQL    READ_FIRST                  ; branch if yes (read blk 1 only)
                                 00AC   346
                                 00AC   347  ;
                                 00AC   348  ; beginning and ending blocks are partially full.
                                 00AC   349  ; read blocks in before update.
                                 00AC   350  ;
                                 00AC   351
                                 00AC   352  GETBLK:
                                 00AC   353
                                 00AC   354  ;
                                 00AC   355  ; save current nrp and set nrp from rp
                                 00AC   356  ;
                                 00AC   357
                                 00AC   358          ASSUME  IRB$W_NRP_OFF EQ IRB$L_NRP_VBN+4
            7E   40 A9   7D      00AC   359          MOVQ    IRB$L_NRP_VBN(R9),-(SP)
      40 A9   48 A9   7D         00B0   360          MOVQ    IRB$L_RP_VBN(R9),IRB$L_NRP_VBN(R9)
                                 00B5   361
                                 00B5   362  ;
                                 00B5   363  ; Now that all the checks have been done to optimize the number of blocks to be
```

```
                        00B5    364 ; read, if this file is being BI journaled, ignore any optimizations and read in
                        00B5    365 ; everything.
                        00B5    366 ;
02 00A0 CA    02    E1  00B5    367         BBC     #IFB$V_BI, IFB$B_JNLFLG(R10), 5$        ; If BI journaling,
              53    D4  00BB    368         CLRL    R3                                      ; Turn off read flags
                        00BD    369 ;
                        00BD    370 ; locate buffer, possibly reading in the current block(s) containing
                        00BD    371 ; the record
                        00BD    372 ;
        FF40'   30      00BD    373 5$:     BSBW    RM$GETBLKNRP
        87 50   E9      00C0    374         BLBC    R0,UPDERR_RSTNRP
                        00C3    375 ;
```

```
                       00C3  377
                       00C3  378  ;
                       00C3  379  ; currrent register contents:
                       00C3  380  ;
                       00C3  381  ;      r11-r8   standard rms
                       00C3  382  ;      r7       end block pointer
                       00C3  383  ;      r6       record data length in bytes
                       00C3  384  ;      r5       record data address in bytes
                       00C3  385  ;      r4       address of current bdb
                       00C3  386  ;      r1       address of current block in buffer
                       00C3  387  ;
                       00C3  388  ; If journaling is enabled for this file, create and write a journal entry
                       00C3  389  ; for the current record.
                       00C3  390  ;
                       00C3  391
       00A0 CA     95  00C3  392         TSTB    IFB$B_JNLFLG(R10)         ; Any journaling enabled?
            15     13  00C7  393         BEQL    UPDATE_REC               ; No, update record in file
            02     BB  00C9  394         PUSHR   #^M<R1S                  ; Yes, save ptr to record destination
            1C     DD  00CB  395         PUSHL   #RJR$_UPDATE             ; Operation to be journaled is a $PUT
  00000000'EF     16  00CD  396         JSB     RM$SEQJNL                ; Journal record
        5E  04    C0  00D3  397         ADDL2   #4, SP                   ; Remove argument from stack
            02     BA  00D6  398         POPR    #^M<R1>                  ; Restore ptr to record destination
        03 50     E8  00D8  399         BLBS    R0, UPDATE_REC           ; If successful, uodate record
          FF6C     31  00DB  400         BRW     UPDERR_RSTNRP            ; Clean up and exit on error
                       00DE  401  UPDATE_REC:
                       00DE  402  UPDATE_REC:
    51  44 A9    C0  00DE  403         ADDL2   IRB$L_NRP_OFF(R9),R1     ; make offset into addr of record
        64 A9    B5  00E2  404         TSTW    IRB$W_ROVRDSZ(R9)        ; any overhead?
            43     13  00E5  405         BEQL    MOVREC                   ; nope
        50 AA    91  00E7  406         CMPB    IFB$B_RFMORG(R10),-      ; stream record?
            04          00EA  407                 #FAB$C_STM
        3D     1E  00EB  408         BGEQU   MOVREC                   ; branch if yes
                       00ED  409
                       00ED  410  ;
                       00ED  411  ; record is either var or vfc
                       00ED  412  ; write out 2 byte binary size field
                       00ED  413  ; (note: it is assumed we always have room for a size field in a block,
                       00ED  414  ; otherwise we would be positioned to the next block already)
                       00ED  415  ;
                       00ED  416
       81  56    B0  00ED  417         MOVW    R6,(R1)+                 ; store size
                       00F0  418
                       00F0  419         ASSUME  <FAB$C_VFC&1> EQ 1
                       00F0  420         ASSUME  <FAB$C_VAR&1> EQ 0
                       00F0  421
    36 50 AA    E9  00F0  422         BLBC    IFB$B_RFMORG(R10),MOVREC ; branch if var rfm
                       00F4  423
                       00F4  424  ; vfc format. store record header
                       00F4  425  ;
                       00F4  426  ;
                       00F4  427
    7E  55    7D  00F4  428         MOVQ    R5,-(SP)                 ; save record addr and size
 56  5F AA    9A  00F7  429         MOVZBL  IFB$B_FSZ(R10),R6        ; get header length
 FE A1  56    A0  00FB  430         ADDW2   R6,-2(R1)                ; increase record size
 55  2C A8    D0  00FF  431         MOVL    RAB$L_RHB(R8),R5         ; get record address
        03     12  0103  432         BNEQ    10$                      ; branch if specified
        55  51    D0  0105  433         MOVL    R1,R5                    ; just copy current header
```

```
                       0108   434                                                       ; (i.e., leaves it unchanged)
                       0108   435  10$:    IFNORD  R6,(R5),ERRRHB,IRB$B_MODE(R9) ; probe it
        008A    30     010F   436          BSBW    BLDREC                        ; move vfc header
     55   8E    7D     0112   437          MOVQ    (SP)+,R5                      ; restore user buffer regs
     0D  50     E9     0115   438          BLBC    R0,UPDERR_BR                  ; get out on error
       FEE5'    30     0118   439          BSBW    RM$PROBEREAD                  ; reprobe user buffer
     07  50     E9     011B   440          BLBC    R0,UPDERR_BR                  ;
         0A     11     011E   441          BRB     MOVREC
                       0120   442  ;
                       0120   443  ; handle errors
                       0120   444  ;
                       0120   445
                       0120   446  ERRRHB: RMSERR  RHB                           ; bad record header buffer
                       0125   447  UPDERR_BR:
        8E     D5      0125   448          TSTL    (SP)+                         ; clean stack
      FF20     31      0127   449          BRW     UPDERR_RSTNRP                 ; exit update
                       012A   450
                       012A   451  ;
                       012A   452  ; now move the data record
                       012A   453  ;
                       012A   454
         70     10     012A   455  MOVREC: BSBB    BLDREC                        ; move rec to buffer
                       012C   456  UPDERR_RSTNRP_1:
     03  50     E8     012C   457          BLBS    R0,5$                         ; get out on error
      FF18     31      012F   458          BRW     UPDERR_RSTNRP
                       0132   459
                       0132   460  ;
                       0132   461  ; Now append DFT to stream format if necessary
                       0132   462  ;
                       0132   463
                       0132   464          ASSUME  FAB$C_STM GT FAB$C_VFC
                       0132   465          ASSUME  <FAB$C_STM+1> EQ FAB$C_STMLF
                       0132   466          ASSUME  <FAB$C_STMLF+1> EQ FAB$C_STMCR
                       0132   467
  00A1 CA   05   93    0132   468  5$:     BITB    #<IFB$M_BI_RECVR!IFB$M_PU_RECVR>, IFB$B_RECVRFLGS(R10)
          1C   12      0137   469          BNEQ    10$                           ; skip if BI journaling
     55   50 AA   9A   0139   470          MOVZBL  IFB$B_RFMORG(R10),R5          ; get format type
          55   04 C2   013D   471          SUBL2   #FAB$C_STM,R5                 ; normalize type
            13   1F    0140   472          BLSSU   10$                           ; not stream format
          64 A9   B5   0142   473          TSTW    IRB$W_ROVHDSZ(R9)             ; anything to add?
            0E   13    0145   474          BEQL    10$                           ; nope
  55  FEB4 CF45  DE    0147   475          MOVAL   W^STM_FMT_DFT[R5],R5          ; point to DFT table
          56   85 9A   014D   476          MOVZBL  (R5)+,R6                      ; get length
            4A   10    0150   477          BSBB    BLDREC                        ; append the DFT
          D7 50   E9   0152   478          BLBC    R0,UPDERR_RSTNRP_1            ; quit on failure
                       0155   479  10$:
                       0155   480  ;
                       0155   481  ; operation now complete.  restore nrp data and return rfa.
                       0155   482  ;
                       0155   483
  74 AA  40 A9   D1    0155   484          CMPL    IRB$L_NRP_VBN(R9),IFB$L_EBK(R10) ; new eof?
          0F   1E      015A   485          BGEQU   CHKEOF                        ; branch if maybe
     40 A9   8E   7D   015C   486  UPDXIT: MOVQ    (SP)+,IRB$L_NRP_VBN(R9)       ; restore nrp
  10 A8   48 A9   7D   0160   487          MOVQ    IRB$L_RP_VBN(R9),RAB$W_RFA(R8)
          62 A9   B4   0165   488          CLRW    IRB$W_CSIZ(R9)                ; indicate no current rec.
         FE95'   31    0168   489          BRW     RMSEXSUC                      ; exit with success
                       016B   490
```

RM1UPDATE   SEQUENTIAL SPECIFIC UPDATE   16-SEP-1984 00:58:37 VAX/VMS Macro V04-00   Page 12
V04-000   RMSUPDATE1 - HIGH LEVEL SEQUENTIAL SUPDA 5-SEP-1984 16:23:54 [RMS.SRC]RM1UPDATE.MAR;1   (8)

E 15

```
                              016B      491  ;
                              016B      492  ;   check to see if this was a random put past current eof and if so
                              016B      493  ;   reset the eof pointer to correspond
                              016B      494  ;
                              016B      495
                              016B      496  CHKEOF:
                              016B      497
                              016B      498  ;
                              016B      499  ;  (note: assumes buff page aligned)
                              016B      500  ;
                              016B      501
                              016B      502          ASSUME   FAB$C_VFC GT FAB$C_VAR
                              016B      503          ASSUME   FAB$C_STM GT FAB$C_VFC
                              016B      504
                   50 AA  91  016B      505          CMPB     IFB$B_RFMORG(R10),-      ; stream format?
                         04  016E      506                   #FAB$C_STM
                         05  1E  016F   507          BGEQU    5$                       ; don't round for stream
                              0171      508  ;
                              0171      509  ;        BITW     #^X1FF,R1                ; Is there anything in this buffer?
                              0171      510  ;        BEQL     30$                      ; if not then we're in an empty buffer.
                              0171      511  ;
                         51  D6  0171   512          INCL     R1                       ; round up offset
                   51    01  AA  0173   513          BICW2    #1,R1
          51  FE00 8F    AA  0176       514  5$:     BICW2    #^XFE00,R1               ; get offset within block
                   03    12  017B       515          BNEQ     10$                      ; branch if not end of block
               40 A9    D6  017D        516          INCL     IRB$L_NRP_VBN(R9)        ; bump nrp
       74 AA    40 A9   D1  0180        517  10$:    CMPL     IRB$L_NRP_VBN(R9),IFB$L_EBK(R10) ; past eof?
                   06    1A  0185       518          BGTRU    20$                      ; branch if yes
          5C AA    51   B1  0187        519          CMPW     R1,IFB$W_FFB(R10)        ; offset past eof offset?
                   CF    1B  018B       520          BLEQU    UPDXIT                   ; branch if not
                              018D      521
       74 AA    40 A9   D0  018D        522  20$:    MOVL     IRB$L_NRP_VBN(R9),IFB$L_EBK(R10) ; reset eof
          5C AA    51   B0  0192        523          MOVW     R1,IFB$W_FFB(R10)
                        0196            524          SSB      #IFB$V_RW_ATTR,(R10)     ; flag attr. rewrite needed
                   CO    11  019A       525          BRB      UPDXIT
                              019C      526
                              019C      527  ;30$:    CMPL     IRB$L_NRP_VBN(R9),IFB$L_EBK(R10) ; Past eof?
                              019C      528  ;        BLEQU    UPDXIT                   ; No, don't update eof.
                              019C      529  ;        MOVL     IRB$L_NRP_VBN(R9),IFB$L_EBK(R10) ; reset eof
                              019C      530  ;        CLRW     IFB$W_FFB(R10)           ; Already know that offset is zero.
                              019C      531  ;        SSB      #IFB$V_RW_ATTR,(R10)     ; falg attr. rewrite needed
                              019C      532  ;        BRB      UPDXIT
```

F 15

RM1UPDATE   SEQUENTIAL SPECIFIC UPDATE   16-SEP-1984 00:58:37 VAX/VMS Macro V04-00   Page 13
V04-000   RMSUPDATE1 - HIGH LEVEL SEQUENTIAL SUPDA 5-SEP-1984 16:23:54 [RMS.SRC]RM1UPDATE.MAR;1   (9)

```
                         019C    534  ;++
                         019C    535  ; BLDREC:        build record subroutine
                         019C    536  ;
                         019C    537  ; this subroutine moves a record from the user record buffer
                         019C    538  ; to the rms i/o buffer, crossing block boundaries as needed.
                         019C    539  ;
                         019C    540  ; Calling sequence:
                         019C    541  ;
                         019C    542  ;         bsbw    bldrec
                         019C    543  ;
                         019C    544  ; Input Parameters:
                         019C    545  ;
                         019C    546  ;         r11     impure area address
                         019C    547  ;         r10     ifab address
                         019C    548  ;         r9      irab address
                         019C    549  ;         r8      rab address
                         019C    550  ;         r7      end of block address + 1
                         019C    551  ;         r6      # of bytes in record
                         019C    552  ;         r5      address of record (source)
                         019C    553  ;         r1      address in rms i/o buffer (destination)
                         019C    554  ;
                         019C    555  ; Implicit Inputs:
                         019C    556  ;
                         019C    557  ;         the contents of the various structures,
                         019C    558  ;         in particular, irb$l_curbdb.
                         019C    559  ;
                         019C    560  ; Output Parameters:
                         019C    561  ;
                         019C    562  ;         r1      address of byte following the moved record
                         019C    563  ;                 in rms i/o buffer
                         019C    564  ;         r0      status code
                         019C    565  ;         r2-r6   destroyed
                         019C    566  ;
                         019C    567  ; Implicit Outputs:
                         019C    568  ;
                         019C    569  ;         bdb$b_flgs - marked dirty
                         019C    570  ;         irb$l_curbdb - updated if block boundary crossed
                         019C    571  ;
                         019C    572  ;         irb$l_nrp_vbn - updated if block boundary crossed
                         019C    573  ;         irb$w_nrp_off - updated if block boundary crossed
                         019C    574  ;
                         019C    575  ; Completion Codes:
                         019C    576  ;
                         019C    577  ;         standard rms.
                         019C    578  ;
                         019C    579  ; Side Effects:
                         019C    580  ;
                         019C    581  ;         if i/o stall occurs will have changed to
                         019C    582  ;         running at ast level; reprobing any non-rab
                         019C    583  ;         user address will be required.
                         019C    584  ;--
                         019C    585
                         019C    586  BLDREC:
50  57  51  C3  019C    587          SUBL3   R1,R7,R0                ; get # bytes left in buffer
    56  50  D1  01A0    588          CMPL    R0,R6                   ; < record size?
        03  1B  01A3    589          BLEQU   20$                     ; branch if so
    50  56  D0  01A5    590          MOVL    R6,R0                   ; no - just use buffer size
```

G 15

```
            56  50  C2  01A8   591 20$:    SUBL2   R0,R6                           ; adjust remaining count
     61  65  50  28  01AB   592         MOVC3   R0,(R5),(R1)                    ; move (partial) record to buffer
        54  20  A9  D0  01AF   593         MOVL    IRB$L_CURBDB(R9),R4             ; get current bdb
     0A A4  03  88  01B3   594         BISB2   #BDB$M_VAL!BDB$M_DRT,BDB$B_FLGS(R4) ; say valid & dirty
            56  D5  01B7   595         TSTL    R6                              ; done?
            16  13  01B9   596         BEQL    40$                             ; branch if yes
        53  51  D1  01BB   597         CMPL    R1,R3                           ; source = destination?
            18  13  01BE   598         BEQL    60$                             ; branch if yes
            51  DD  01C0   599         PUSHL   R1                              ; save source addr
            1A  10  01C2   600         BSBB    CHNGBf                          ; move to next buffer
        55 8ED0  01C4   601         POPL    R5                              ; restore source addr
        0D 50  E9  01C7   602         BLBC    R0,50$                          ; get out on error
        FE33' 30  01CA   603         BSBW    RM$PROBEREAD                    ; reprobe user buffer
        CC 50  E8  01CD   604         BLBS    R0,BLDREC                       ; and go again if no error
            05  01D0   605         RSB
               01D1   606
               01D1   607 ;
               01D1   608 ; move to buffer is complete
               01D1   609 ;
               01D1   610
               01D1   611 40$:;   CMPL    R7,R3                           ; Have we exactly filled the buffer?
               01D1   612 ;       BEQL    55$                             ; If equal then we have, force it out.
        51  53  D0  01D1   613         MOVL    R3,R1                           ; next byte pointer to correct reg.
            01D4   614         RMS$SUC
            05  01D7   615 50$:    RSB
               01D8   616
               01D8   617 ;
               01D8   618 ; force the current buffer to be written out.
               01D8   619 ;
               01D8   620
               01D8   621 ;55$:   BSBB    CHNGBf                          ; move to next buffer
               01D8   622 ;       BLBC    R0,50$                          ; get out on error
               01D8   623 ;       BSBW    RM$PROBEREAD                    ; reprobe user buffer
               01D8   624 ;       RSB
               01D8   625
               01D8   626 ;
               01D8   627 ; since the source and destination pointers are equal, this is a
               01D8   628 ; copy of the existing vfc header.  read the next buffer and simply
               01D8   629 ; bump the pointer in the block as the vfc header is definitely not
               01D8   630 ; longer than the new buffer.
               01D8   631 ;
               01D8   632
        04  10  01D8   633 60$:    BSBB    CHNGBf                          ; read in next block buffer
     51  56  C0  01DA   634         ADDL2   R6,R1                           ; bump buffer addr past rest
               01DD   635                                                 ; of header
            05  01DD   636         RSB
               01DE   637
               01DE   638 ;
               01DE   639 ; change buffer/block subroutine
               01DE   640 ;
               01DE   641 ; calls rm$nxtblk1 subroutine with r3 set to read in the next block
               01DE   642 ; unless the block will be completely filled by the record, in which
               01DE   643 ; case no read is required.
               01DE   644 ; all other inputs and outputs same as for rm$nxtblk1
               01DE   645 ;
               01DE   646
        53  01  D0  01DE   647 CHNGBF: MOVL    #1,R3                           ; flag no read required
```

RM1UPDATE
V04-000

H 15

SEQUENTIAL SPECIFIC UPDATE   16-SEP-1984 00:58:37 VAX/VMS Macro V04-00   Page 15
RMSUPDATE1 - HIGH LEVEL SEQUENTIAL SUPDA 5-SEP-1984 16:23:54 [RMS.SRC]RM1UPDATE.MAR;1   (9)

```
        48 AA  56  B1  01E1  648              CMPW    R6,IFB$L_DEVBUFSIZ(R10) ; will block be filled?
               15  1E  01E5  649              BGEQU   CHNGBF1                 ; branch if yes
    50  40 A9  01  C1  01E7  650              ADDL3   #1,IRB$L_NRP_VBN(R9),R0 ; compute next vbn
        74 AA  50  D1  01EC  651              CMPL    R0,IFB$L_EBK(R10)       ; past eof?
               08  1F  01F0  652              BLSSU   10$                     ; branch if not (must read)
               08  1A  01F2  653              BGTRU   CHNGBF1                 ; branch if yes (no read)
                       01F4  654
                       01F4  655      ;
                       01F4  656      ; in the eof block  -  check for read required
                       01F4  657      ;
                       01F4  658
        5C AA  56  B1  01F4  659              CMPW    R6,IFB$W_FFB(R10)       ; any bytes that won't be overwritten?
               02  1E  01F8  660              BGEQU   CHNGBF1                 ; branch if none (no read)
               33  D4  01FA  661  10$:        CLRL    R3                      ; flag read required
                       01FC  662  CHNGBF1:
        FE01'  31      01FC  663              BRW     RM$NXTBLK1              ; go read next block
                       01FF  664
                       01FF  665
                       01FF  666              .END
```

```
$$.PSECT_EP              = 00000000              RAB$W_RFA                = 00000010
$$RMSTEST               = 0000001A              READ_FIRST               00000066 R      01
$$RMS_PBUGCHK           = 00000010              RJR$_UPDATE              = 0000001C
$$RMS_TBUGCHK           = 00000008              RM$E_RMS                 ********    X   01
$$RMS_UMODE             = 00000004              RM$EXSUC                 ********    X   01
BDB$B_FLGS              = 0000000A              RM$GETBLKNRP             ********    X   01
BDB$M_DRT               = 00000002              RM$NXTBLK1               ********    X   01
BDB$M_VAL               = 00000001              RM$PROBEREAD             ********    X   01
BLDREC                    0000019C R      01    RM$PUTSETUP1             ********    X   01
CHKEND                    0000006A R      01    RM$PUT_UNIT_REC          ********    X   01
CHKEOF                    0000016B R      01    RM$SEQJNL                ********    X   01
CHNGBF                    000001DE R      01    RM$UPDATE1               0000000C RG     01
CHNGBF1                   000001FC R      01    RM$UPDATE_ALT            00000079 RG     01
CR                      = 0000000D              RMS$_CUR                 = 000184B4
DEV$V_RND               = 0000001C              RMS$_IOP                 = 00018574
ERRCUR                    00000043 R      01    RMS$_RHB                 = 0001866C
ERRRHB                    00000120 R      01    RMS$_RSZ                 = 000186A4
ERRRSZ                    0000003C R      01    STM_FMT_DFT              00000000 R      01
FAB$C_STM               = 00000004              TPT$L_UPDATE1            ********    X   01
FAB$C_STMCR             = 00000006              UPDATE                   00000087 R      01
FAB$C_STMLF             = 00000005              UPDATE_REC               000000DE R      01
FAB$C_VAR               = 00000002              UPDERR                   0000004E R      01
FAB$C_VFC               = 00000003              UPDERR_BR                00000125 R      01
GETBLK                    000000AC R      01    UPDERR_RSTNRP            0000004A R      01
IFB$B_FSZ               = 0000005F              UPDERR_RSTNRP_1          0000012C R      01
IFB$B_JNLFLG            = 000000A0              UPDXIT                   0000015C R      01
IFB$B_RECVRFLGS         = 000000A1
IFB$B_RFMORG            = 00000050
IFB$L_DEVBUFSIZ         = 00000048
IFB$L_EBK               = 00000074
IFB$L_PRIM_DEV          = 00000000
IFB$M_BI_RECVR          = 00000004
IFB$M_RU_RECVR          = 00000001
IFB$V_BI                = 00000002
IFB$V_DAP               = 0000003E
IFB$V_RW_ATTR           = 00000034
IFB$W_FFB               = 0000005C
IRB$B_MBC               = 00000055
IRB$B_MODE              = 0000000A
IRB$L_CURBDB            = 00000020
IRB$L_NRP_OFF           = 00000044
IRB$L_NRP_VBN           = 00000040
IRB$L_RP_VBN            = 00000048
IRB$V_FIND_LAST         = 00000025
IRB$V_UPDATE            = 00000033
IRB$W_CSIZ              = 00000062
IRB$W_NRP_OFF           = 00000044
IRB$W_ROVADSZ           = 00000064
IRB$W_RP_OFF            = 0000004C
LF                      = 0000000A
MOVREC                    0000012A R      01
NOFIT                     00000061 R      01
NOREAD                    0000006E R      01
NOREAD1                   00000074 R      01
NTUPD                     0000005A R      01
PIO$A_TRACE              ********    X   01
RAB$L_RHB               = 0000002C
```

```
                                    +-----------------+
                                    ! Psect synopsis  !
                                    +-----------------+

PSECT name                  Allocation          PSECT No.   Attributes
----------                  ----------          ---------   ----------
.  ABS  .                   00000000  (      0.) 00 (    0.)  NOPIC  USR  CON  ABS  LCL  NOSHR  NOEXE  NORD  NOWRT  NOVEC  BYTE
RMS$RMS1                    000001FF  (    511.) 01 (    1.)    PIC  USR  CON  REL  GBL  NOSHR    EXE   RD   NOWRT  NOVEC  BYTE
$ABS$                       00000000  (      0.) 02 (    2.)  NOPIC  USR  CON  ABS  LCL  NOSHR    EXE   RD     WRT  NOVEC  BYTE

                            +---------------------------+
                            ! Performance indicators !
                            +---------------------------+

Phase                   Page faults   CPU Time        Elapsed Time
-----                   -----------   --------        ------------
Initialization                  35    00:00:00.10     00:00:01.32
Command processing             123    00:00:00.75     00:00:06.18
Pass 1                         350    00:00:11.46     00:00:25.73
Symbol table sort                0    00:00:01.53     00:00:02.62
Pass 2                         121    00:00:02.62     00:00:06.99
Symbol table output             11    00:00:00.10     00:00:00.54
Psect synopsis output            2    00:00:00.03     00:00:00.03
Cross-reference output           0    00:00:00.00     00:00:00.00
Assembler run totals           644    00:00:16.60     00:00:43.41
```

The working set limit was 1500 pages.
65733 bytes (129 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1226 non-local and 16 local symbols.
666 source lines were read in Pass 1, producing 14 object records in Pass 2.
25 pages of virtual memory were used to define 24 macros.

```
                            +----------------------------+
                            ! Macro library statistics !
                            +----------------------------+

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                     14
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                      1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   5
TOTALS (all libraries)                             20
```

1336 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RM1UPDATE/OBJ=OBJ$:RM1UPDATE MSRC$:RM1UPDATE/UPDATE=(ENH$:RM1UPDATE)+EXECML$/LIB+LIB$:RMS/LIB

RM!PUTREC
LIS

RM!PUTSET
LIS

RM!UPDATE
LIS

RM!NXTBLK   RM!PUTBLD   RM!RELBLK   RM!SEQXFR
LIS   LIS   LIS   LIS

RM!PUT   RM2CONN
LIS   LIS

RM!OPEN   RM!WTLST
LIS   LIS

                                                           RM!STMFMT
                                                           LIS